Path Navigation for Robot Using Matlab

Prof. A. G. Andurkar[1], Ms. Rupali Tankar[2], Ms. Suvarna Patil[3]
Assistant prof., E&TC, Govt.college of engineering , Jalgaon, India [1]
M.Tech, E&TC, Govt.college of engineering, Jalgaon, India [2]
M.Tech, E&TC, Govt.college of engineering, Jalgaon, India [3]

Abstract- Path navigation using fuzzy logic controller and trajectory prediction table is to drive a robot in the dynamic environment to a target position, without collision. This path navigation method consists of static navigation method and dynamic path planning. The static navigation used to avoid the static obstacles by using fuzzy logic controller, which contains four sensor input and two output variables. If the robot detects moving obstacles, the robot can recognize the velocity and moving direction of each obstacle and generate the Trajectory Prediction Table to predict the obstacles' future trajectory. If the trajectory prediction table which reveals that the robot will collide with an obstacle, the dynamic path planning will find a new collision free path to avoid the obstacle by waiting strategy or detouring strategy.

A lot of research work has been carried out in order to solve this problem. In order to navigate successfully in an unknown or partially known environment, the mobile robots should be able to extract the necessary surrounding information from the environment using sensor input, use their built-in knowledge for perception and to take the action required to plan a feasible path for collision free motion and to reach the goal.

Keywords-*fuzzy logic controller,trajectory prediction table,matlab*

*****

I. INTRODUCTION

Path navigation is important to robots, which is to find an optimal collision-free path from a starting point to a target in a given environment according to some criteria such as distance, time or energy [7]. There are two main types of path planning: Global Path Planning, which encompasses all the acquired knowledge of a robot (i.e, environmental information is known) to reach a goal; and Local Navigation, which is the process of using only the robot's currently sensed information (environmental information is unknown or partially unknown) [10]. A number of studies have been done for global path planning, such as visibility graph methods, grid method, freespace method [1]. There have been many studies for local path planning, such as neural networks, artificial potential field method, and fuzzy logic algorithm [1]. There are two types of obstacles in the environment: static and dynamic obstacles. There are so many methods already developed for static obstacle avoidance. Thus, recent interests aim at dynamic obstacle-avoidance including avoidance of both static and dynamic obstacles using fuzzy logic and trajectory prediction table [7][10][4].

Robot navigation is a fundamental problem in robotics. Navigation related to mobile robot is the ability of finding a collision free path from its starting position to the goal position by avoiding the obstacles. Moreover the selected path should be optimized i.e. having smallest possible distance and minimum number of turns to make sure that least amount of energy and time are used by the robot in roaming from starting point to its target

II. LITERATURE REVIEW

A. Potential Field Method

Andrews and Hogan (1983) and Khatib (1985) have been suggested imaginary forces acting on a robot in such a way that the robot has been attracted by its target and obstacle exerts repulsive forces on it. Therefore resultant force governs the following direction and speed of travel. Borenstein and Koren (1991) have presented a systematic overview and discussed about the inherent drawbacks of potential field methods (PFMs). No passage between closely spaced obstacles (when two obstacles are presents very near to each other, then the robot may repelled away from the obstacles). Oscillations in the presence of obstacles and Oscillations in the narrow passages. Moreover they have developed a new method namely vector field histogram (VHF) method which gives smooth, non-oscillatory motion and can be used for fast obstacle avoidance[1]. Cosio and Castaneda (2004) have presented a new scheme for autonomous navigation of mobile robot based on improved artificial potential field and genetic algorithm (GA). They have used the concept of multiple auxiliary attraction points so that the robot can overcome the trap situation and avoid the closely spaced obstacles. The intensity of attractive and repulsive forces along with the position parameter of the auxiliary attraction point has been optimized by the GA. Huang (2009) Addressed a new

technique, derived from the classical potential filed method, for the velocity planning of a mobile robot in a dynamic environment where the target and obstacles are moving. The method ensures that the robot successfully tracks the moving target by avoiding the obstacles along its path, by providing the direction and the speed of the robot. Olunloyo and Ayomoh (2009) have presented a path planning and obstacle avoidance approach for a mobile robot in a partially known 2-D environment.

B. Neuro-Fuzzy Approach

The difficulties of constructing correct fuzzy rule base are overcome by using neural network-based approaches [36]. Online learning method has been adopted and dynamic rule generation is developed using neural fuzzy. The optimization of rule construction is done in this paper. This has been found to be good when applied to wall following control of omni directional robot. The neuro-fuzzy algorithm for reactive robot is proposed and developed [10]. This technique can effectively deal with imprecise coordinate conflicts among multi-behavior contexts. RAM-based neural network is used to supervise the FLC. The benefit in using RAM based neural network approach is that it requires very less memory and less computation time maintaining the ability to handle imprecise and complex data.

Heuristic method of reasoning is adopted instead of analytical solution. The proposed behavior-based navigation strategy using fuzzy rules has major benefits compared to the mathematical model. Even though the fuzzy-based reasoning is easy to implement but for multibehavior coordination, it's difficult to find out optimized rules to overcome all sorts of destructions. Artificial neural network is combined along with fuzzy logic reasoning method, and this maps the inputs and outputs in optimized way. The fuzzy logic and back propagation neural network been used for road traffic signs detection and classification [5]. Fuzzy logic has been used for the sign detection and classification, for back propagation neural network technique is used to display the right task. The purpose of the paper is to make walking safer and easier. The system is divided in to three different stages-first stage detection and improving raw sign image, second is shape analysis with continuous thinning algorithms and the image coding algorithm, finally, image recognition and decision by fuzzy logic and back propagation neural network technique to display the right task.

The application of mobile robot navigation with obstacle avoidance has been done using polynomial neural network. This polynomial neural network is built from some selected starting location to reach the goal. The efficient technique based on associative retrieval is applied to robot to follow minimal cost polynomial path. This has advantage of interpolating capability with moderate size of data space. Use of new adaptation block for mobile robot to learn new behavioral actions and scripts based-soft computing techniques [10].

C. Fuzzy Logic Based Navigation

Barret, Benreguieg, and Maaref (1997) have proposed a sensor-based navigation algorithm, combines two types of obstacle avoidance behaviours, each for the convex obstacles and the concave ones. To avoid the convex obstacles the navigator uses either fuzzy tuned artificial potential field (FTAPF) method or a behavioural agent, however an automatically online wall-following system using a neuro-fuzzy structure has been designed for the concave one. Xu (2000) has proposed a virtual target approach for resolving the limit cycle problem in navigation of a behaviour-based mobile robot. The real target has been switched to a virtual location so that robot can navigate according to the virtual target until it detects the opening. The efficiency and effectiveness of the refined fuzzy behaviour-based navigation are demonstrated by means of both simulation and physical experiments. Aguirre Eugenio and Gonzalez Antonio (2000) dealt with a hybrid deliberative-reactive architecture for mobile robot navigation for integrating planning and reactive control, and attention is focused on the design, coordination and fusion of the elementary behaviours. Saade and Khatib (2003) have developed a data-driven fuzzy approach to provide a general framework for solving the Dynamic motion problem (DMP) problem of a mobile robot under some constraints[7]. The main advantage of the current approach over recent fuzzy-genetic one is that the robot can navigate successfully in the presence of moving obstacles and independently of the number of these obstacles. The proposed approach has also reveals the reduction in the travel time. The proposed algorithm has shown good results as compared to ANFIS on robot trajectory in terms of their length and the time required by the robot to reach the goal. The superiority of the new algorithm can be helpful in building fuzzy models without any compulsion of planting effort in gaining accurate and enormous number of data points. Li and Hseng (2003) have designed and implemented a new fuzzy controller for a car-like mobile robot (CLMR) that holds autonomous garage-parking and parallel-parking capacity by using real time image processing. The system consists of a host computer, a communication module, a CLMR, and a vision system. Fuzzy garage parking control (FGPC) and fuzzy parallel parking control (FPPC) have been used in order to control the steering angle of the CLMR. Cuesta *et al.* (2003) have presented a new method for the intelligent control of the nonholonomic vehicles. Fuzzy perception has been directly used, both in design of each reactive behaviour and solving the problem of behaviour combination in order to implement a fuzzy behaviour based

control architecture. The capabilities of the control system have been improved by considering teleoperation and planned behaviour, together with their combination with reactive ones. Experimental results have shown the robustness of the suggested technique. Abdessemed Foudil, Benmahammed Khier, and Monacelli Eric (2004) have used the fuzzy logic controller in the development of complete navigation procedure of a mobile robot in a messy environment. An evolutionary algorithm has been implemented in order to solve the problem of extracting the IF-THEN rule base. The validity of the proposed method has been demonstrated through simulation results. Demirli and Molhim (2004) have presented a new fuzzy logic based approach for dynamic localization of mobile robots. The proposed approach uses sonar data obtained from a ring of sonar sensors mounted around the robot. The angular uncertainty and radial imprecision of sonar data are modelled by possibility distributions.

The information received from the adjacent sonar sensors are united, which helps in the reduction in the uncertainty in sonar impressions. In the beginning a local fuzzy map has been constructed with help of reduced models of uncertainty, and then fitted to the given global map of the environment to identify robot's location. This fit offers either a unique fuzzy location or multiple candidate fuzzy locations. Since the coordinates (x, y) and orientation of the identified locations are represented by possibility distribution, these locations are referred to as fuzzy locations. To reduce the number of candidate locations, a new set of candidate fuzzy location is obtained by moved the robot to a new position. By considering the robot's movement, a set of hypothesized locations is identified from the old set of candidate locations. The hypothesized locations are matched with the new candidate locations and the candidates with low degree of match are eliminated. This process is continued until a unique location is obtained. The matching process is performed by using the fuzzy pattern matching technique. The proposed method is implemented on a Nomad 200 robot and the results are reported. Parhi (2005) has described a fuzzy logic based control system for the navigation of multiple mobile robots in a cluttered environment, such that the robots do not collide to each other. For this he has used fuzzy logic controller to combine the fuzzy rules in order to direct the steering of the robot to avoid the obstacles present in its path. Moreover Petri Net model has been used by implementing crisp rules to avoid the collision between the different mobile robots. Simulation and test results validate the system functions by enabling the robots to reach their goal without hitting the static obstacles or colliding with other robots[7]. Fatmi *et al.* (2006) have demonstrated a successful way of constructing the navigation task in order to deal with problem of autonomous navigation of mobile robot. Issues of individual behaviour design and action coordination of the behaviours were addressed using fuzzy logic. They have designed the individual behaviours like goal reaching; emergency situation, obstacle avoidance, and wall following are presented using fuzzy if-then rule base. Moreover they have introduced a coordination technique to overcome the problem of activation of several behaviours independently or/and simultaneously. Mendez and Madrigal (2007) have proposed a user adaptive fuzzy based navigation system for the autonomous navigation of mobile robot in unknown environments. They have tested their system in a pioneer mobile robot and on a robotic wheel chair, equipped with PLS laser sensor for the detection of obstacles and odometry sensors for the localization of the robot and the goal positions. The proposed system has a learning algorithm that can quickly adapt to different users[5][7]. They have found that the proposed system takes 90% less computation time for the task as compared to others reactive control tested in the same platform for the previous system. Hassanzadeh, Ghadiri, and Dalayimilan (2008) have used a simple fuzzy controller for obstacle avoidance of mobile robot navigation.

Summary

The use of fuzzy logic for local navigation of robots is a much discussed topic in literature. Various implementations have been shown to be effective and efficient. The ability of fuzzy techniques to deal with imprecise data allows for smooth trajectory execution while their low computational complexity allows them to react quickly to dynamic environments without the need to alter the robot's end-to-end path. Because algorithms based on fuzzy logic depend on sensory data to make navigational corrections, they are essentially reactive in nature. This quality can elicit suboptimal behaviors including shortsightedness and the local minima problem. A knowledge of longer range obstacles as from long-range sensory instruments or conventional model-based planning algorithms could allow the reactive behavior of a fuzzy controller to make quick trajectory adjustments while still approximating the most preferred path. Despite the limitations and non-deterministic nature of fuzzy algorithms, they have been shown to produce robust local navigation control for robots in unknown and noisy environments. The literature reviewed here shows that, with modest differential complexity, fuzzy algorithms can be used practically in both 2D and 3D environments as well as with industrial manipulators. As such, they represent a complement to rather than a replacement for conventional motion planners across a wide range of applications where real time, autonomous obstacle avoidance is needed.

III. SYSTEM DESCRIPTION

A. Concept

Table 1. List of Rules for Static Robot Navigation

Input variable				Output Variable	
Left	*Front*	*Right*	*Angle* [b]	$\Delta\theta$ [a]	*v*
Near	Near	Near	Right	NB	Slow
Near	Near	Medium	–	NS	Slow
Near	Near	Far	–	NB	Slow
Near	Medium	Near	–	ZE	Slow
Near	Medium	Medium	–	NS	Medium
Near	Medium	Far	–	NS	Medium
Near	Far	Near	–	ZE	Medium
Near	Far	Medium	–	ZE	Medium
Near	Far	Far	–	NS	Medium
Medium	Near	Near	–	PB	Slow
Medium	Near	Medium	–	NB	Slow
Medium	Near	Far	–	NB	Slow
Medium	Medium	Near	–	PS	Medium
Medium	Medium	Medium	–	ZE	Slow
Medium	Medium	Far	–	NS	Medium
Medium	Far	Near	–	ZE	Medium
Medium	Far	Medium	–	ZE	Fast
Medium	Far	Far	–	ZE	Fast
Far	Near	Near	–	PB	Slow
Far	Near	Medium	–	PB	Slow
Far	Near	Far	–	NB	Slow
Far	Medium	Near	–	PS	Medium
Far	Medium	Medium	–	PS	Medium
Far	Medium	Far	–	NS	Medium
Far	Far	Near	–	PS	Medium
Far	Far	Medium	–	ZE	Fast
Far	Far	Far	–	ZE	Fast
Near	Near	Near	Left	PB	Slow

a) NB: Negative Big, NS: Negative Small, ZE: Zero, PS: Positive Small, PB: Positive Big.

b) θ^o: Angle indicates the included angle between the robot and target.

Table 2. shows a trajectory prediction table. In this, A robotic path planning method containing the robot navigation for static obstacles and dynamic path planning for moving obstacles, as shown in Fig.1. The task of the robot is to move from a start position to target position without collision. We assume that the static and dynamic obstacles are unknown. There are two kinds of sensors for the robot. The sonar sensors with a maximum of 130 cm detect the stationary obstacles in left, front, and right of the robot. The laser range finder with a maximum of 250 cm used to detect the omnidirectional moving obstacles. As listed Line 1 of Algorithm of the robotic path planning, the robot moves forward until it reaches the target. The robot firstly detects if there exists any dynamic obstacle around it. If the dynamic obstacles are recognized, the algorithm will generate the Trajectory Prediction Table, e.g., TABLE 2. From the current time step, e.g., *t*, to several future time steps, e.g., *t* + 9, we predict the positions of the robot and detected obstacles, and estimate if the robot has a collision or not at each time step. This table will be used to plan a short-term path to avoid the moving obstacle. In Lines 6-11, if there is no dynamic obstacle detected by the robot's sensor within a range, i.e., 250 cm in this paper, or there exists dynamic obstacles but without collision, the robot will move toward to the target directly if it does not detect any static obstacle as well. Otherwise, the robot have to avoid the stationary obstacles by using the fuzzy logic control. In Lines 12-14, if the robot detects dynamic obstacles which will collide with that robot, the dynamic path planning, by using Trajectory Prediction Table, is used to find a short-term path to avoid the obstacle.

- *Algorithm of the proposed robotic path planning.*

1 **while** (the robot does not reach the target),
2 Detect whether exist any dynamic obstacle around the robot;
3 **if** dynamic obstacles are detected **then**
4 Generate the Trajectory Prediction Table;
5 **end if**
6 **if** (no dynamic obstacle) or (exist obstacles but no collision) **then**
7 **if** no static obstacle around the robot **then**
8 The robot directly moves toward the target;
9 **else**
10 Do the *static navigation* by fuzzy logic control;
11 **end if**
12 **else**
13 Do *dynamic path planning* using Trajectory Prediction Table;
14 **end if**
15 **end while**

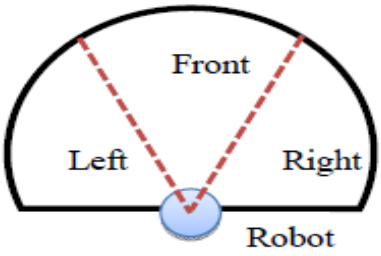

Fig 1. Algorithm of the Proposed Robotic Path Planning.

B. Fuzzy Logic for Static Navigation

Fig 1. illustrates the sensing range of sonar sensors of the robot which is inspired by [10], and the sensing range is divided into three sectors, *Left*, *Front*, and *Right*. We apply the fuzzy logic controller to derive the robot avoiding the static obstacles. The input variables are: the distances between the static obstacle and the **Left**, **Front**, **Right** sides of robot (the ranges are [0, 130]cm), and the **included angle** between the robot and the target (its range is [0°, 180°]). The output variables $\Delta\theta$ and v denote the increment of the robot's steering angle (whose range is [-120°, 120°]) and the velocity of the robot (whose range is [10, 40] cm/sec). We assume that each discrete time step corresponds to one second. The membership functions of input variables Left, Front, and Right are same as shown in Fig.2.

 ISSN: 2321-8169
46 – 53

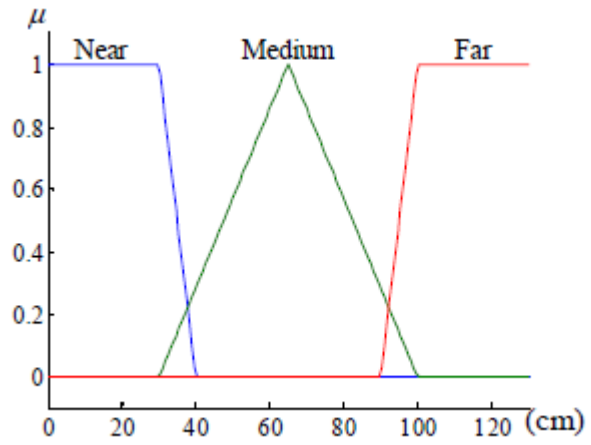

Fig.2 The Membership Function of Input Variable **Left**, **Front**, and **Right**.

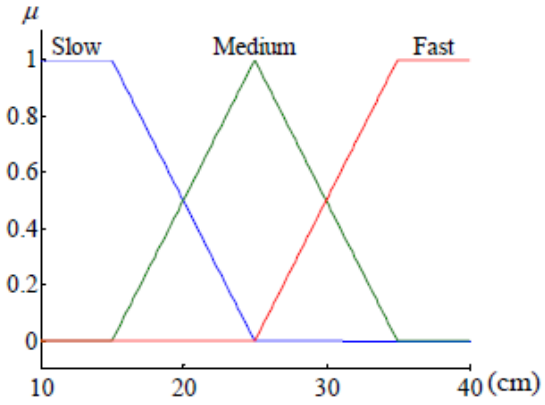

Figure.3 The membership function of output variable v.

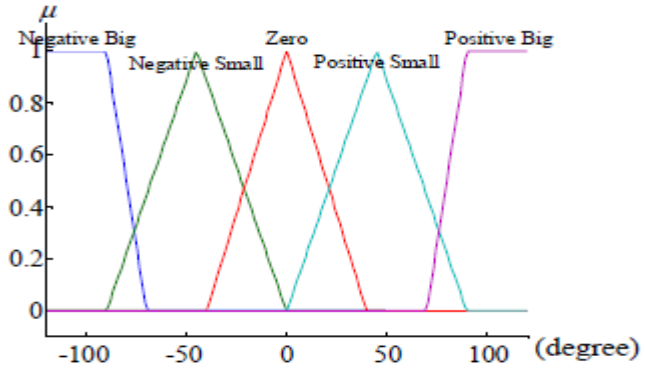

Fig.4 The membership function of output variable Δθ.

We can use the crisp membership function for input variable **included angle**. If the degree of that angle is less than 90, it belongs to "Right" linguistic term; otherwise, it belongs to "Left". The membership functions of v and Δθ are shown in Figs.3 and 4, respectively[13].

C. Dynamic Path Planning

If the robot detects dynamic obstacles, the Trajectory Prediction Table will be generated to predict the trajectories of moving obstacles and the robot in next several time steps.

- *Algorithm of Dynamic Path Planning*

1 **for** *time* = 1 to *nTraj*, // *nTraj*: number of time steps in trajectory table
2 **for** each dynamic obstacle,
3 **if** the collisions start at *time* = *tc* **then**
4 *tS* = *tc* – 1;
5 *tG* = the first time step without collision after *time* = *tc*;
6 *newPath* = **FindNewPath**(*trajTable*, *tS*, *tG*);
7 **end if**
8 break; // exit the inner **for** loop
9 **end**
10 **if** *newPath* is not empty **then** break; **end if**
11 **end**
12 *path* = *path* (1 → *tS* – 1) ∪ *newPath*;
13 **procedure** FindNewPath(*trajTable*, *tS*, *tG*)
14 *rTraj* = robot's trajectory in *trajTable*;
15 *waitTime* = *tG* – *tS* – 1; //waiting time: no. of time steps should wait
16 **if** (*waitTime* < hWait **or** *tG* = *nTraj* + 1) **and** no collision during
waiting **then**
17 *newPath* = *rTraj*(1 → *tS* – 1) ∪ [repeat *rTraj*(*tS*) *waitTime* + 1];
18 **else**
19 **if** *tS* > 1 **then** *tS'* = *tS* – 1; **end if**
20 *rS'* = *rTraj*(*tS'*); *rG* = *rTraj*(*tG*);
21 *time* = *tS'*;
22 *newPath* = *rTraj*(1 → *tS'*);
23 **while** (*rG* is not reached),
24 *rNext* = **GetSuccessors**(*newPath*(*time*));
25 **for** each *rNext*,
26 **if** a collision occurs in the moving to *rNext* **then**
27 *g*(*rNext*) = ∞; *f*(*rNext*) = ∞;
28 **else**
29 *g*(*rNext*) = distance from *newPath*(*time*) to *rNext*;
30 *h*(*rNext*) = distance from *rNext* to *rG*;
31 *f*(*rNext*) = *g*(*rNext*) + *h*(*rNext*);
32 **end if**
33 **end for**
34 *time* =*time* + 1; *newPath*(*time*) = the *rNext* with minimal *f*;
35 **end while**
36 **end if**
37 return *newPath*;
38 **procedure** GetSuccessors(*r*)
39 θ = included angle between robot's current position *r* and target *rG*;
40 φ = the moving angle of obstacle;
41 **if** φ < 180° **then**
42 **if** θ < φ + 180° **then** φLow = φ;
43 **else** φLow = φ + 180°;
44 **end if**
45 **else**
46 **if** θ < φ **then** φLow = φ − 180°;
47 **else** φLow = φ;
48 **end if**
49 **end if**
50 ***v*** = {vSlow, vMedium, vFast}; // robot's velocity
51 ϕ = {φLow, φLow + 10°, φLow + 20°, …, φLow + 180°}; // robot's direction
52 r = (rx, ry); rNext = null;

 ISSN: 2321-8169
46 – 53

53 **for** i = 1 to 3,
54 **for** j = 1 to 19,
55 rx
' = rx + **v**(i)cos(ϕ (j)); // *x* coordinate
56 *ry*
' = ry + v(i)sin(ϕ (j)); // *y* coordinate
57 rNext = rNext ∪ (rx
', ry
');
58 **end for**
59 **end for**
60 return rNext;

According to the Trajectory Prediction Table (such as Table 2), if the robot will collide with moving obstacle in the future, the proposed dynamic path planning will be used to find a short-term path for avoiding the obstacle. There are two strategies to avoid dynamic obstacle in this paper, i.e., waiting (degree) μ or detouring around the moving obstacle.

Table 2. An Illustration of Trajectory Prediction Table

Time Step	(*x*, *y*) Coordinates			Colliding Detection	
	Robot	*Obstacle 1*	*Obstacle 2*	*Collision 1*[a]	*Collision 2*
1	$(r_x, r_y)^{(1)}$	$(o_{1x}, o_{1y})^{(1)}$	$(o_{2x}, o_{2y})^{(1)}$	No	No
2	$(r_x, r_y)^{(2)}$	$(o_{1x}, o_{1y})^{(2)}$	$(o_{2x}, o_{2y})^{(2)}$	No	No
3	$(r_x, r_y)^{(3)}$	$(o_{1x}, o_{1y})^{(3)}$	$(o_{2x}, o_{2y})^{(3)}$	No	No
4	$(r_x, r_y)^{(4)}$	$(o_{1x}, o_{1y})^{(4)}$	$(o_{2x}, o_{2y})^{(4)}$	No	No
5	$(r_x, r_y)^{(5)}$	$(o_{1x}, o_{1y})^{(5)}$	$(o_{2x}, o_{2y})^{(5)}$	No	No
6	$(r_x, r_y)^{(6)}$	$(o_{1x}, o_{1y})^{(6)}$	$(o_{2x}, o_{2y})^{(6)}$	Yes	No
7	$(r_x, r_y)^{(7)}$	$(o_{1x}, o_{1y})^{(7)}$	$(o_{2x}, o_{2y})^{(7)}$	Yes	No
8	$(r_x, r_y)^{(8)}$	$(o_{1x}, o_{1y})^{(8)}$	$(o_{2x}, o_{2y})^{(8)}$	Yes	No
9	$(r_x, r_y)^{(9)}$	$(o_{1x}, o_{1y})^{(9)}$	$(o_{2x}, o_{2y})^{(9)}$	No	No

Collision 1 indicates whether a collision occurs with obstacle 1 at time step *t*.

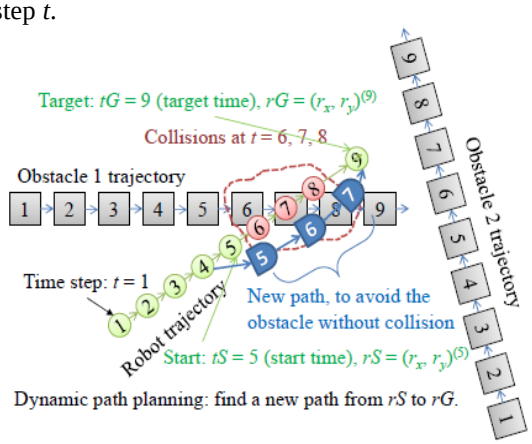

Fig 5. An Illustration of Dynamic Path Planning.

The algorithm of dynamic path planning is listed, and an illustration is shown in Fig.5. At time step t = 1, the robot detects two dynamic obstacles, and thus the Trajectory Prediction Table, as shown in Table 2, is generated to predict the obstacles' positions at each time step (see Lines 2 to 5, algorithm Algorithm of the proposed robotic path planning). The numbers located within the circle and rectangular represent the time step. *nTraj* (see Line 1, Algorithm of Dynamic Path Planning) denotes the number of time steps in trajectory table, and it is the maximal moving steps of the robot (one time step indicates the robot moving one step forward) within the maximal sensing range of laser range finder, i.e., 250 cm. As shown in Table 2. and Fig. 5, *nTraj* = 9. At time steps 6, 7, and 8, the robot will collide with obstacle 1. Lines 2 and 9 in Algorithm of Dynamic Path Planning are to find which obstacle will collide with robot, and that is the obstacle 1. As listed from Lines 3 to 7 in Algorithm of Dynamic Path Planning, if the collisions start at time step 6 (*tc* = 6), as shown in Fig. 5, and then we let *tS* = 6 – 1 = 5. The collisions end at time step 8, and thus *tG* = 8 + 1 = 9. *rS* and *rG* are the start and target positions of the robot at time step *tS* and *tG*, respectively. In Line 6 of Algorithm of Dynamic Path Planning, *newPath* = **FindNewPath**(*trajTable*, *tS*, *tG*) means that use the *trajTable* (Trajectory Prediction Table), *tS*, and *tG* as input parameters for FindNewPath procedure to discover a short-term path, storing to *newPath*, to avoid obstacle 1.We assume that the robot will have only one collision at most. Once we obtain the *newPath* by using FindNewPath procedure, exit **for**-loop to go to Line 12. Finally, the planned path is to merge the new path (*newPath*) with the first half of the original path which starts from time step 1 to time step *tS* – 1 (see Line 12,)[13].

The core of dynamic path planning is the FindNewPath procedure as shown in Lines 13 to 37. Firstly, the robot has to choose the avoiding strategy, *waiting* strategy or *detouring* strategy. The premise of Line 16 means: The waiting time (wait until the obstacle moves away) is less than *hWait* (a tolerable threshold for wait; *hWait* = 2 in this paper) or *tG* = *nTraj* + 1 (it means a collision occurs at the last time step, i.e., *nTraj*), and there is no any collision if the robot waits at time step *tS* (the robot stays on *rS*, e.g., (*rx*, *ry*)(5) in Fig.5). If the premise is satisfied, the robot will adopt the waiting strategy to wait *waitTime* steps; otherwise, it will use the detouring strategy. In Fig.4, the *waitTime* = 9 – 5 – 1 = 3. Since *waitTime* > 2 (*hWait* = 2), the robot will plan a path to detour the obstacle. In Line 17 of Algorithm of Dynamic Path Planning, *rTraj*(1 → *tS* – 1) specifies the first *tS* – 1 coordinates of *rTraj*. For an example of Table 2., if the robot adopt the waiting strategy, *tS* = 5 and *waitTime* = 3. *rTraj*(1 → *tS* – 1) = 4 ( ) 1 (,) *tt rx ry* = ∗ , and [repeat *rTraj*(*tS*) *waitTime* + 1] = 4 (5) *t* 1 (*rx* , *ry*) = ∗ . Thus, *newPath* = 4 ( )1 (,) *t t rx ry* = ∗ ∗ [4 (5) *t* 1 (*rx* , *ry*) = ∗].

Lines 19 to 35 of Algorithm of Dynamic Path Planning are the process for finding a path to detour the obstacle. Before starting the main procedure shown in Lines 23 to 35, we have to settle the parameters. The action of line 19 let the robot move back one step to give a wider moving space. In Lines 19, 20, and 22, for an example of Table 2. and Fig. 5, *tS'* = 5 – 1 = 4, *rS'* = (*rx*, *ry*)(4), *rG* = (*rx*, *ry*)(9), and *newPath* = 4 ( ) 1 (,) *t t rx ry* = ∗ .

Similar to the A* algorithm [4], Lines 23 to 35 intend to obtain a shortest path, from *rS'* to *rG*, without collision. As shown in Fig.3, the moving positions of new path are labeled by the flowchart symbol, "delay." The new trajectory consists of three coordinates occurring at time steps 5, 6, and 7, denoted by (*r'x*, *r'y*)(5), (*r'x*, *r'y*)(6), and (*r'x*, *r'y*)(7). In order to obtain the short and safety path, the robot has to move faster. Thus, the new path only requires four moving steps, but the original path requires five steps from *t* = 4 to *t* = 9. Within the **while**-loop, from Lines 23 to 35, the robot will choose the next position *rNext* to move with minimal cost *f*(*rNext*) = *g*(*rNext*) + *h*(*rNext*) until it reaches or very close to the target *rG*.

g(*rNext*) and *h*(*rNext*) are the distances defined in Lines 29 and 30. If the robot collides with an obstacle in the moving from current position to *rNext*, we let *g*(*rNext*) = ∞ and *f*(*rNext*) = ∞ (infeasible).
Line 24: *rNext* = **GetSuccessors**(*newPath*(*time*)) is to get all next possible positions given the current (*x*, *y*) coordinate *newPath*(*time*) of the robot by using the GetSuccessors procedure, shown in Lines 38 to 50. Lines 39 to 49 are to obtain φ*Low* which is the smallest steering angle of the robot among all angles and is parallel to the moving angle of obstacle. For example of Fig.5, the moving angle of obstacle 1 is 0°. To get the next position to move, there are three possible velocities (shown in Line 50) and 19 possible steering angles (shown in Line 51) for the robot to choose. Lines 53 to 59 are to calculate all 3 × 19 (= 57) combinations, (*x*, *y*) coordinates, to store into *rNext*[10][13[9].

D. Simulation Code

```
clc
clear all

% The starting direction is found from the start point and the
end point
% The inputs are three sensors for detection of obstacle which
fuzzified
% on the basis of far, close or very close and the output is the
navigation
% i.e. straight, soft right, hard right, soft left, hard left.

% The distance of the obstacle is sensed from the sensor and
depends on the
% current position and orientaion of the moving bot.

range=[0 40];% sensor data distance range for simulation it
can be the pixel distance
vclose=[8 0];%very close
close=[5 15];%close
far=[5 25];%far

fuzzy_img_struct = newfis('path');%Fuzzy inference system

%% fuzzy variable for input
fuzzy_img_struct =
addvar(fuzzy_img_struct,'input','SR',range);
fuzzy_img_struct =
addmf(fuzzy_img_struct,'input',1,'low','gaussmf',vclose);
fuzzy_img_struct =
addmf(fuzzy_img_struct,'input',1,'medium','gaussmf',close);
fuzzy_img_struct =
addmf(fuzzy_img_struct,'input',1,'high','gaussmf',far);
figure,plotmf(fuzzy_img_struct,'input',1), title('Sensor Right');

fuzzy_img_struct =
addvar(fuzzy_img_struct,'input','SS',range);
fuzzy_img_struct =
addmf(fuzzy_img_struct,'input',2,'low','gaussmf',vclose);
fuzzy_img_struct =
addmf(fuzzy_img_struct,'input',2,'medium','gaussmf',close);
fuzzy_img_struct =
addmf(fuzzy_img_struct,'input',2,'high','gaussmf',far);
figure,plotmf(fuzzy_img_struct,'input',2), title('Sensor
Straight');

fuzzy_img_struct =
addvar(fuzzy_img_struct,'input','SL',range);
fuzzy_img_struct =
addmf(fuzzy_img_struct,'input',3,'low','gaussmf',vclose);
fuzzy_img_struct =
addmf(fuzzy_img_struct,'input',3,'medium','gaussmf',close);
fuzzy_img_struct =
addmf(fuzzy_img_struct,'input',3,'high','gaussmf',far);
figure,plotmf(fuzzy_img_struct,'input',3), title('Sensor Left');

%% fuzzy variable for output
%In degrees of navigation
range=[-120 120];
m1val=[-80 -120];
m2val=[-90 -20];
m3val=[-40 40];
m4val=[20 90];
m5val=[60 120];

%In terms of membership normalized for gaussmf
t=max(range-min(range));
range1=(range-min(range))/max(range-min(range));
m1val=(m1val-min(range))/t;
m2val=(m2val-min(range))/t;
m3val=(m3val-min(range))/t;
m4val=(m4val-min(range))/t;
m5val=(m5val-min(range))/t;

fuzzy_img_struct =
addvar(fuzzy_img_struct,'output','Dir',range1);
fuzzy_img_struct = addmf(fuzzy_img_struct,'output',1,'Hard
Right','gaussmf',m1val);
fuzzy_img_struct = addmf(fuzzy_img_struct,'output',1,'Soft
Right','gaussmf',m2val);
fuzzy_img_struct =
addmf(fuzzy_img_struct,'output',1,'Straight','gaussmf',m3val);
fuzzy_img_struct = addmf(fuzzy_img_struct,'output',1,'Soft
Left','gaussmf',m4val);
fuzzy_img_struct = addmf(fuzzy_img_struct,'output',1,'Hard
Left','gaussmf',m5val);
figure,plotmf(fuzzy_img_struct,'output',1), title('Direction');
```

REFERENCES

[1] Koren, Y. Borenstein, J. (1991),'Potential Field Methods and their Inherent Limitations for Mobile Robot Navigation', *In the Proceedings of the IEEE Conference on Robotics and Automation, April 7-12, 1991. Sacramento, California. pp. 1398-1404.*
[2] Huang, L. (2009), 'Velocity Planning for a Mobile Robot to Track a Moving Target - A Potential Field Approach', *Robotics and Autonomous Systems, 57, 55-63.*
[3] L. Sun, Y. Luo, X. Ding and L. Wu, "Path planning and obstacle avoidance for mobile robots in a dynamic environment," *The Open Automation and Control Systems Journal*, vol. 6, pp. 77-83, 2014..
[4] M. Faisal, K. Al-Mutib, R. Hedjar, H. Mathkour, M. Alsulaiman, and E. Mattar, "Multi modules fuzzy logic for mobile robots navigation and obstacle avoidance in unknown indoor dynamic environment," in *Proceedings of 2013 International Conference on Systems, Control and Informatics*, pp. 371-379, 2013.
[5] M. K. Singh, D. R. Parhi, S. Bhowmik, and S. K. Kashyap, "Intelligent controller for mobile robot: Fuzzy logic approach.," in *Proceedings of 12th International Conference of International*

Association for Computer Methods and Advances in Geomechanics, pp. 1-6, 2008.

[6] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *Proceedings of 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5628-5635, 2011.

[7] M. Wang and J. N. K. Liu, "Fuzzy logic-based real-time robot navigation in unknown environment with dead ends," *Robotics and Autonomous Systems*, vol. 56, pp. 625–643, 2008.

[8] N. T. Thanh and N. V. Afzulpurkar, "Dynamic path planning for a mobile robot using image processing," Journal of Computer Science and Cybernectics, vol. 24, no. 4, pp.358-373, 2008.

[9] P. Raja and S. Pugazhenthi, "Path planning for a mobile robot in dynamic environments," *International Journal of the Physical Sciences*, vol. 6, no. 20, pp. 4721-4731, 2011.

[10] R. Kala, A. Shukla, and R. Tiwari, "Dynamic environment robot path planning using hierarchical evolutionary algorithms," Cybernetics and Systems: An International Journal, vol. 41, no. 6, pp. 435-454, 2010.

[11] Y. Lu, X. Huo, O. Arslan, and P. Tsiotras, "Incremental multi-scale search algorithm for dynamic path planning with low worst-case complexity," *IEEE Transactions on Systems, Man, and Cybernetics— Part B: Cybernetics, vol. 41, no. 6, pp. 1556-1570, 2011.*

[12] Yang, Simon X. and Meng, Max (2000), 'An Efficient Neural Network approach to Dynamic Robot Motion Planning', *Neural Networks, 13, 143–148.*

[13] Z. Wu and L. Feng, "Obstacle prediction–based dynamic path planning for a mobile robot," *International Journal of Advancements in Computing Technology, vol. 4, no. 3, pp. 118-124, 2012.Z.*.